KAIST

CS 380

Introduction to Computer Graphics

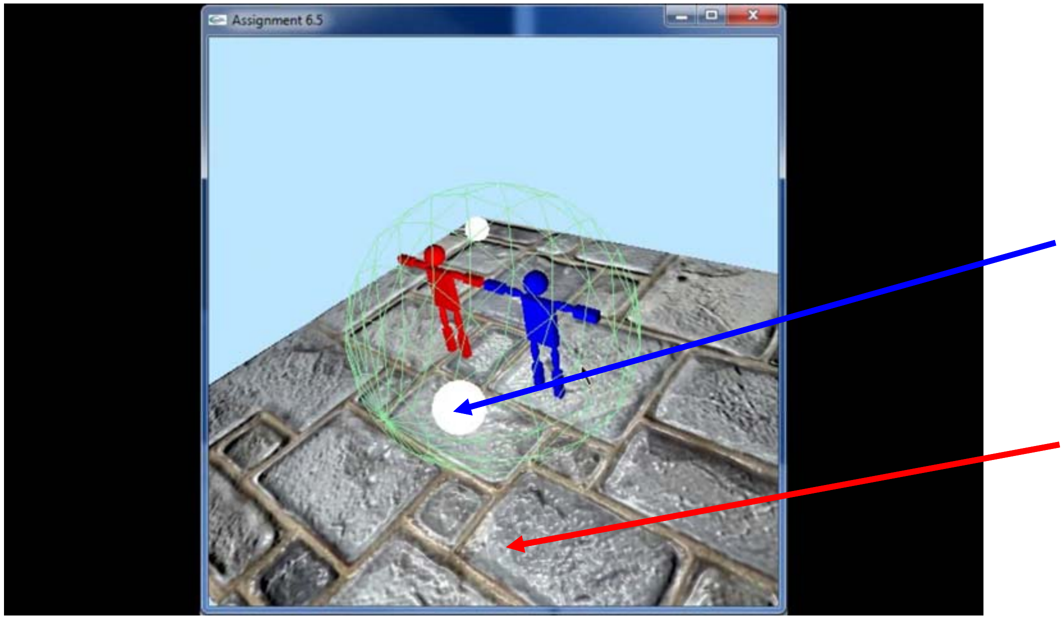
LAB (7)

2018.05.14~17

VISUAL COMPUTING Lab

Overview

- Material Infrastructure
  - Multiple shaders per one frame
- Bump mapping
  - Normal map

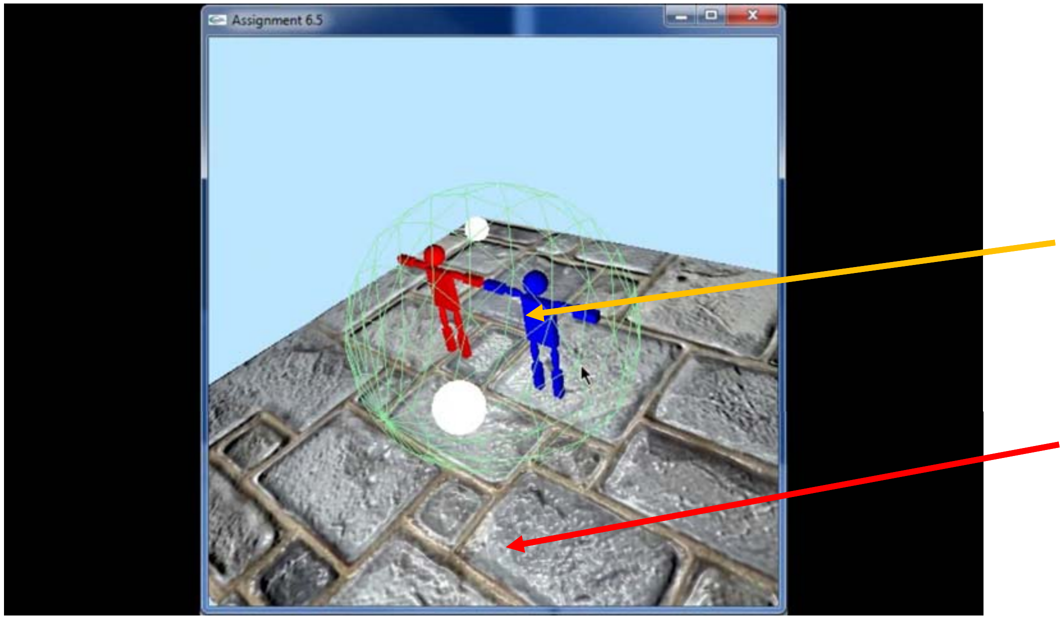


Light

Bump

Multiple Shaders

- Each shader has own ***uniform*** variables
- Different GLSL shader do not know about the values of each other’s uniform variables



Diffuse material

Bump material

Transferring Uniform Value

Uniform.h

- Uniform: dictionary mapping from name to value

Uniforms.put(the name of the variable in the shader, ***the actual value*** *)*

Types: float, int, matrix4, shared_ptr <Texture>, …

Drawer
Picker
SgShapeNode

E.g.)

```
// Suppose uniforms is of type Uniforms, and m is of type Matrix4
uniforms.put("uProjection", m);
// Suppose light is of type Cvec3
uniforms.put("uLight", light);
// Set uColor variable to red
uniforms.put("uColor", Cvec3(1, 0, 0));
// You can even chain the put, since put returns the object itself
uniforms.put("a", 1)
.put("b", 10)
.put("c", Cvec2(1, 2));
```

RenderStates

- *RenderStates*: A subset of OpenGL state
- State does not immediately take effect in OpenGL
- The state will be applied when you call the member function: ***apply()***
- Useful for multi-shader case

E.g.)

```
RenderStates r1;
r1.enable(GL_BLEND);
r1.apply();
```

Geometry & Texture

- Complex types of geometry and texture

- Geometry
  - GeometryPN: position and normal
  - GeometryPNTBX: position, normal, tangent, binormal, and texture coordinate
- Texture
  - ImageTexutre

Material

- ***Material*** contains three parts
  - Shared pointer
    - GLSL shader program used
  - Uniforms
    - accessible through *getUniforms()*
  - RenderStates
    - accessible through *getRenderStates ()*

- Member function
  - *.draw(geometry, extraUniforms)*

E.g.)

```
sendModelViewNormalMatrix(uniforms, MVM, normalMatrix(MVM));
g_arcballMat->draw(*g_sphere, uniforms);
```

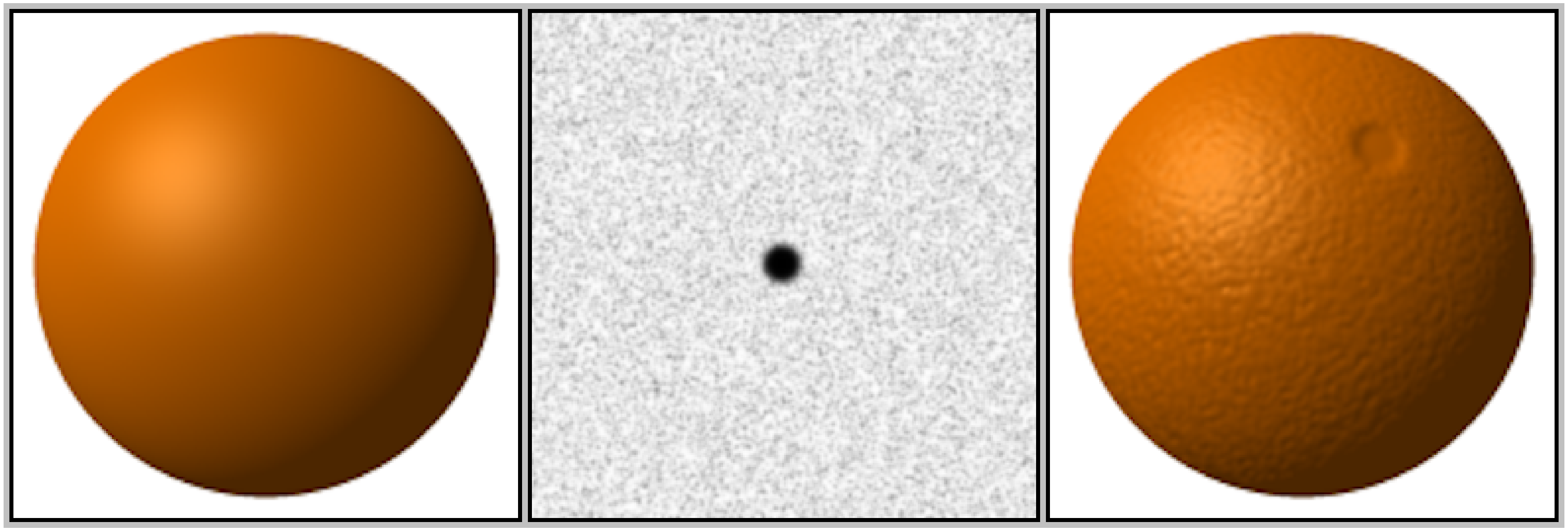
Scene Graph & Material

- Each *SgGeometryShapeNode* has own “Material”

- The robots: diffuse color
- The arcball: wireframe and solid color
- The ground: texture

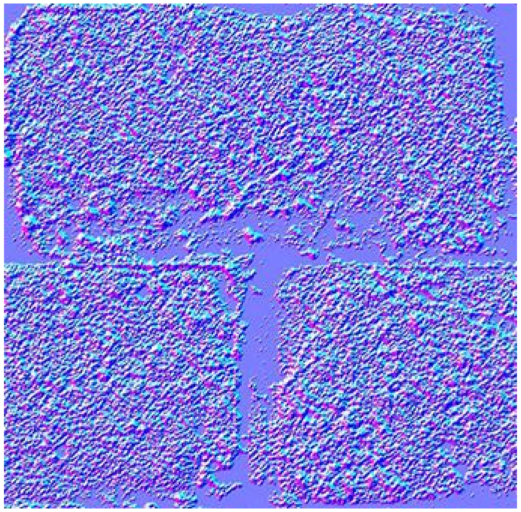
Bump Mapping

- Simulating the bumps on the surface



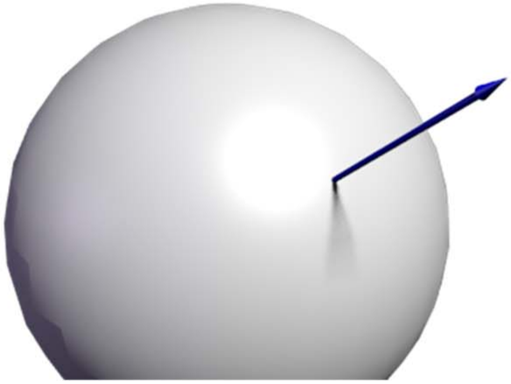
- Instead of changing the geometry itself,
- Modify the surface normal to simulate bumps

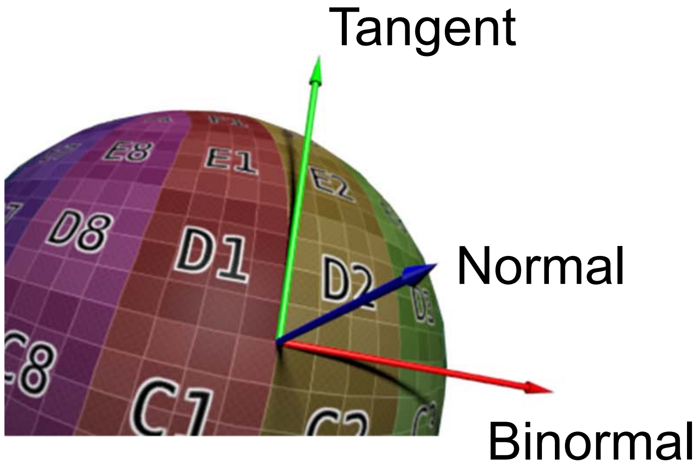
Bump Mapping



Normal map

$$\mathbf{n} = [n_r, n_g, n_b, 0]^t$$





- Normal map defined w.r.t. the tangent frame
- Object frame: $\vec{b}^t = \vec{e}^t M$
- Tangent frame: $T(1:3,1:3) = [\text{tangent,binormal,normal}]$ $\vec{t}^t = \vec{b}^t T$
- New normal: $M^{-t} T \mathbf{n}$

TODO

- Task 1 : Read the pdf file and understand the material infrastructure.
  - Then, migrate the code.

- Task 2: Bump Mapping.
  - Make the lights. (two lights which pickable and movable)
  - Wrtie some GLSL code.